

A quick guide to using LArSoft

Karl Warburton with help from Tingjun Yang

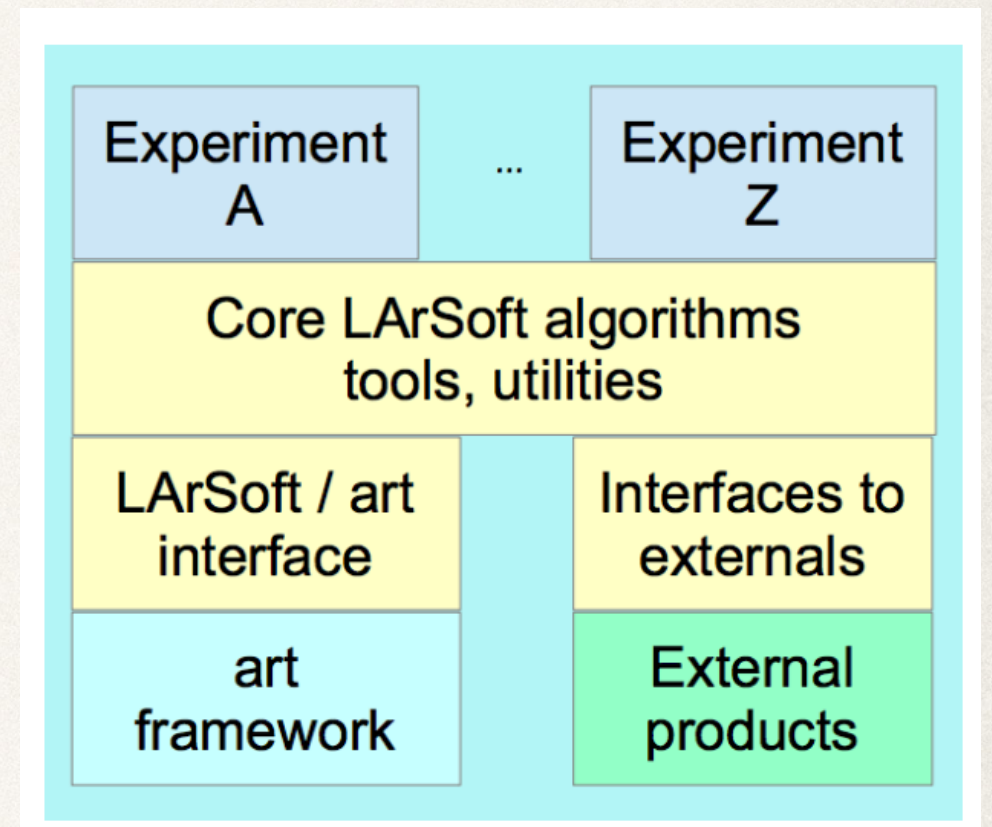
02/02/2016

Guide to this guide

- ❖ A lot of the information shown here is taken from;
 - ❖ The dunetpc cheat sheet, which is [here](#).
 - ❖ The LArSoft guide, which is [here](#).
 - ❖ The 35 ton getting start guide, which is [here](#).
 - ❖ The LArSoft concepts webpage, which is [here](#).
 - ❖ An art/LArSoft course in June '15, which is [here](#).
 - ❖ Very in-depth talks, I'm skimming over some stuff which they cover really well here.
- ❖ LArSoft relies on the art framework which was developed by the Fermilab scientific computing division for intensity frontier experiments.
 - ❖ A useful (though HUGE) handbook to help use art can be found [here](#).

Structure of LArSoft

- ❖ “The LArSoft software (the body of code) is designed to work for all planned and running liquid argon experiments at Fermilab”
- ❖ Experiment specific code is held in experiment repositories, such as specific geometry files and intricate analysis code.
- ❖ The code for the general reconstruction, analysis, data type declarations, generators and event displays are held in ‘common’ repositories.
- ❖ The ‘common’ repositories collectively are called the LArSoft suite.



The LArSoft suite

Name	Description
larcore	Low level utilities and functions e.g. Geometry services
lardata	Data products and other common data structures
larevt	Low level algorithm code that use data products
larsim	Simulation code
larreco	Primary reconstruction
larana	Secondary reconstruction/analysis e.g. PID
lareventdisplay	LArSoft based event display
larpandora	LArSoft interface to the pandora reconstruction package
larexamples	Placeholder for examples

- ❖ All packages can be checked out individually or as a whole – more on this in a moment...
- ❖ All code within a repository is within a subdirectory of the same name eg larcore/larcore.

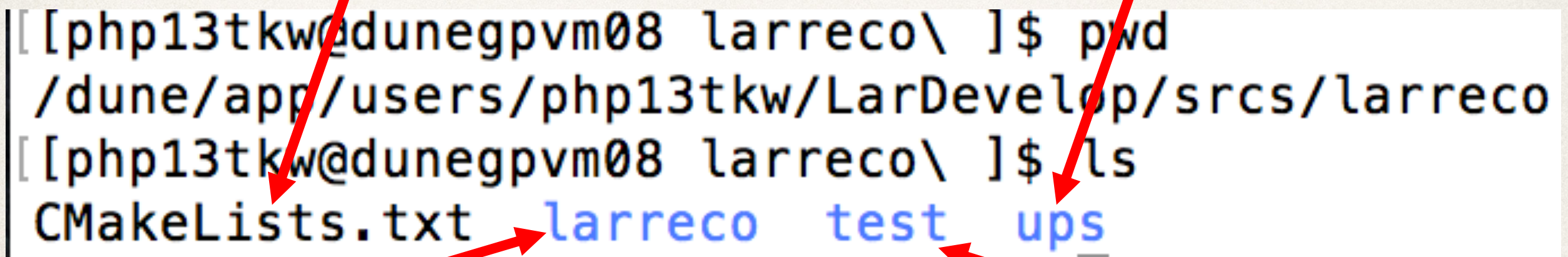
What is in a given repository?

- ❖ For examples sake we will take larreco – the repo with the reconstruction in it.

A file used by the build system to execute certain steps.

A directory for configuration files, dependency lists, etc

```
[php13tkw@dunegpvm08 larreco\]$ pwd
/dune/app/users/php13tkw/LarDevelop/srcs/larreco
[php13tkw@dunegpvm08 larreco\]$ ls
CMakeLists.txt  larreco  test  ups
```



Source code directories under a single directory, so all the hit finding and track making is in here – the code we're interested in.

A directory for unit and integration tests organized by source directory

What is in a given repository?

```
[[php13tkw@dunegpvm08 larreco\ ]$ cd larreco/
[[php13tkw@dunegpvm08 larreco\ ]$ ls
CMakeLists.txt  DirOfGamma  Genfit      MComp      ShowerFinder  TrackFinder  WireCell
ClusterFinder  EventFinder HitFinder   RecoAlg    SpacePointFinder  VertexFinder
```

- ❖ Separate directories for different aspects of reconstruction.
 - ❖ Hits
 - ❖ Cluster finding
 - ❖ Space point finding
 - ❖ Track finding
 - ❖ Showers
- ❖ Within these directories is the code for the processes, for example in HitFinder there are loads of hit finding algorithms.

```
[[php13tkw@dunegpvm08 larreco\ ]$ cd HitFinder/
[[php13tkw@dunegpvm08 HitFinder\ ]$ ls
APAHitFinder_module.cc  HitAnaAlg.h  RFFHitFinderAlg.h  hitfindermodules.fcl
CMakeLists.txt          HitAnaModule_module.cc  RFFHitFinder_module.cc  hitfindermodules_argoneut.fcl
DisambigCheater_module.cc  HitCheater_module.cc  RFFHitFitter.cxx  hitfindermodules_bo.fcl
DumpHits_module.cc        HitFilterAlg.cxx  RFFHitFitter.h  hitfindermodules_dune.fcl.example
FFTHitFinder_module.cc    HitFilterAlg.h  RawHitFinder_module.cc  hitfindermodules_jp250L.fcl
GausHitFinderAna_module.cc  HitFinderAna_module.cc  RegionAboveThresholdFinder.cxx  hitfindermodules_microboone.fcl
GausHitFinder_module.cc    HitFinder_module.cc  RegionAboveThresholdFinder.h  mchitmodules.fcl
GaussianEliminationAlg.cxx  MCHitAnaExample_module.cc  TTHitFinder_module.cc
GaussianEliminationAlg.h    MCHitFinder_module.cc  dump_hits.fcl
HitAnaAlg.cxx              RFFHitFinderAlg.cxx  hitana.fcl
```


How to get your hands on repositories

- ❖ The repositories are all git projects, meaning that you 'pull' them using git commands.
- ❖ From your srcs directory you type:
 - ❖ `mrbs g larreco`
- ❖ Should you want to use a specific version (that isn't the version of develop), then you type:
 - ❖ `mrbs g -t v05_14_00 larreco`
- ❖ I am skipping quite a few steps here, but will come back to this later...

Steps to understand LArSoft

- ❖ Just to get started we need to know some things about:
 - ❖ git – analagous to svn which is used in NOvA
 - ❖ How versions are controlled in LArSoft
 - ❖ How different repositories talk to each other
 - ❖ How to setup your local environment
 - ❖ Producers vs Analyzers vs Algorithms

A rough and ready explanation of git

- ❖ Git allows multiple people to use and update a common item(s) in parallel. It can be used for files (eg friends have spoken highly of it for theses), smallish packages (an event display in LArSoft was originally in this form), or big packages eg LArSoft.
- ❖ It allows versioning control, so if you want to revert to a previous state of a file it's easy!
- ❖ Whenever someone changes something they explain what they are changing with a 'commit message'
- ❖ You can have multiple branches so if you have a base file which you need to manipulate in two different ways you could:
 - ❖ Manipulate in way A on branch Karl_A
 - ❖ Manipulate in way B on branch Karl_B

A rough and ready explanation of git 2

- ❖ When you have a git 'project' you have a master branch, which is where files are stored for production (in LArSoft we never touch this branch and all work is done on develop or other branches)
- ❖ You can list all the branches with:
 - ❖ `git branch -a`
- ❖ You can move to a branch which already exists with:
 - ❖ `git checkout feature/Karl_OldBranch.`
- ❖ You can make a new branch with:
 - ❖ `git flow feature start Karl_NewBranch`
- ❖ To let other people see and use this branch you also need to do:
 - ❖ `git flow feature publish Karl_NewBranch`

A rough and ready explanation of git 3

- ❖ ****Do lots and lots of coding**** To push all of that fancy code do:
 - ❖ `git add <file path within directory>`
 - ❖ `git commit -m "I did loads of things!"`
 - ❖ `git push`
- ❖ Whilst working on your feature branch develop is likely to change, so you'll need to merge develop into your feature branch
 - ❖ `git checkout develop`
 - ❖ `git pull`
 - ❖ `git checkout feature/Karl_NewBranch`
 - ❖ `git merge develop`

A rough and ready explanation of git 4

- ❖ Merging your code into develop
 - ❖ `git checkout develop`
 - ❖ `git merge feature / Karl_NewBranch`
- ❖ When the project is finished you have two choices
 - ❖ Delete the feature branch locally
 - ❖ `git branch --delete feature / Karl_NewBranch`
 - ❖ Delete the feature branch completely
 - ❖ `git push origin --delete feature / Karl_NewBranch`
- ❖ Merge your feature branch into develop
 - ❖ `git flow feature finish`
 - ❖ `git push`

Resolving easy git conflicts

- ❖ Someone will invariably change a file you have changed at some point and a 'git pull' or 'git merge' will fail.

```
[[php13tkw@dunegpvm08 dunetpc\ ]$ git pull
remote: Counting objects: 52, done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 29 (delta 20), reused 0 (delta 0)
Unpacking objects: 100% (29/29), done.
From ssh://cdcvns.fnal.gov/cvs/projects/dunetpc
   56c2a55..69c493c  develop      -> origin/develop
   bef7ee8..5ae6262  feature/rnd_PmaModuleSplit -> origin/feature/rnd_PmaModuleSplit
   4a2aee1..4fb703e  feature/wallbank_APACrossingMuons -> origin/feature/wallbank_APACrossingMuons
Updating 56c2a55..69c493c
error: Your local changes to the following files would be overwritten by merge:
       fcl/dunefd/gen/single/prodsingle_dune10ktdphase.fcl
Please, commit your changes or stash them before you can merge.
Aborting
```

- ❖ Luckily though there are tools git has which you can use to fix this.

Resolving easy git conflicts

- ❖ If you have both changed a file but in different places, it is easy.
- ❖ You can then use git stash
- ❖ Stores your changes to a temporary location.
- ❖ You can then pull develop and do git stash pop to merge in your changes.

```
[php13tkw@dunegpvm08 dunetpc\ ]$ git stash
Saved working directory and index state WIP on develop: 56c2a55 Changing CMakeLists to use art_make.
HEAD is now at 56c2a55 Changing CMakeLists to use art_make.
[php13tkw@dunegpvm08 dunetpc\ ]$ git pull
Updating 56c2a55..69c493c
Fast-forward
 dune/Utilities/signalservices_dune.fcl | 2 +-
 fcl/dunefd/gen/single/prodsingle_dune10ktdphase.fcl | 1 +
 fcl/dunefd/reco/standard_reco_dune10ktdphase.fcl | 3 +++
 3 files changed, 5 insertions(+), 1 deletion(-)
[php13tkw@dunegpvm08 dunetpc\ ]$ git pull
Already up-to-date.
[php13tkw@dunegpvm08 dunetpc\ ]$ git stash pop
Auto-merging fcl/dunefd/gen/single/prodsingle_dune10ktdphase.fcl
On branch develop
Your branch is up-to-date with 'origin/develop'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   fcl/dunefd/gen/single/prodsingle_dune10ktdphase.fcl

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        ups/product_deps.bak

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (b5f151faeb2dd5580eea76927589304155b34247)
[php13tkw@dunegpvm08 dunetpc\ ]$ git status
On branch develop
Your branch is up-to-date with 'origin/develop'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   fcl/dunefd/gen/single/prodsingle_dune10ktdphase.fcl

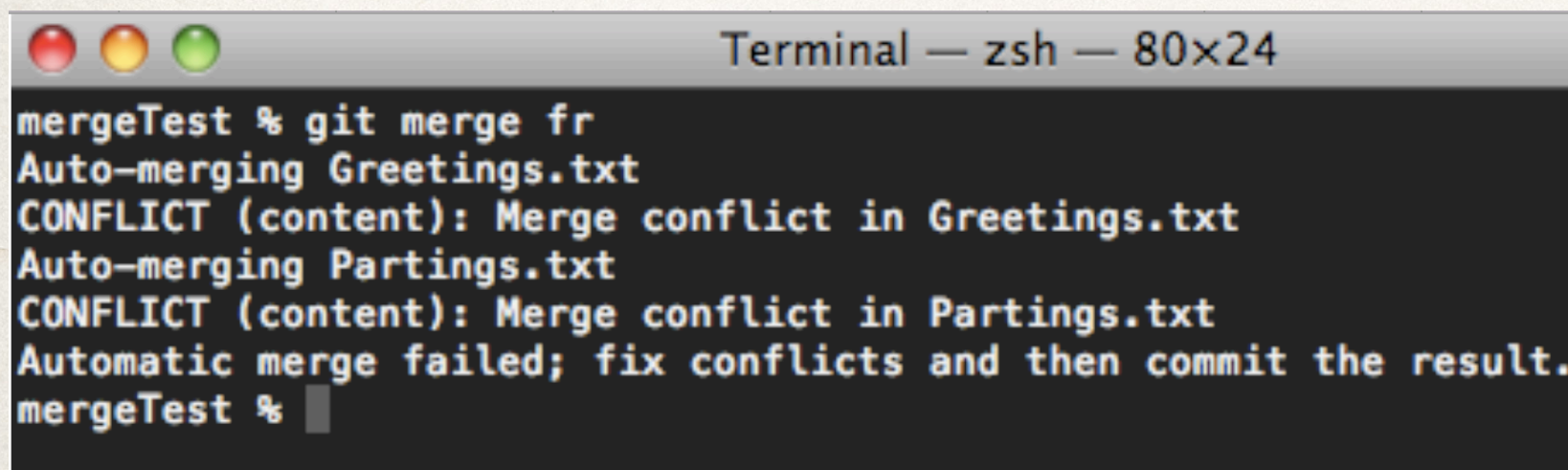
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        ups/product_deps.bak

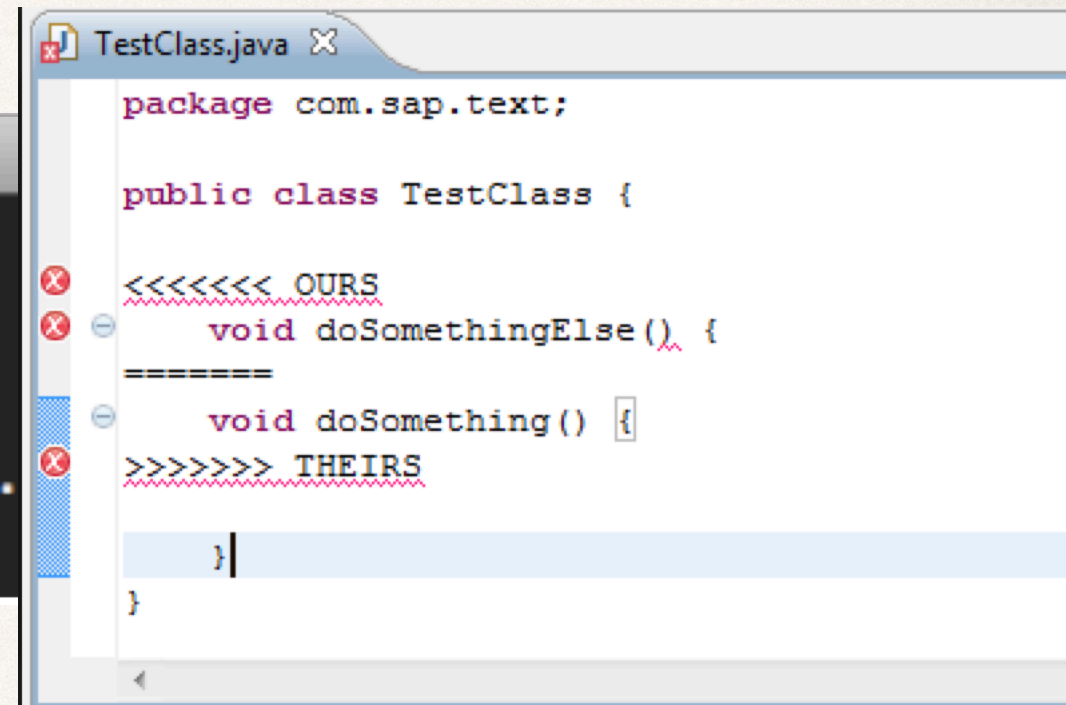
no changes added to commit (use "git add" and/or "git commit -a")
```


Resolving harder git conflicts

- ❖ Or harder if you both changed the same line, and then when you do git stash pop you will get an message saying that you must fix the conflicts.



```
Terminal — zsh — 80x24
mergeTest % git merge fr
Auto-merging Greetings.txt
CONFLICT (content): Merge conflict in Greetings.txt
Auto-merging Partings.txt
CONFLICT (content): Merge conflict in Partings.txt
Automatic merge failed; fix conflicts and then commit the result.
mergeTest %
```



```
TestClass.java
package com.sap.text;

public class TestClass {
    <<<<<<< OURS
    void doSomethingElse() {
    =====
    void doSomething() {
    >>>>>>> THEIRS
    }
}
```

- ❖ You fix conflicts by deciding what the relevant should be in regions indicated by “<<<<<<” and “>>>>>>”
- ❖ Once you have done this you then do
 - ❖ git commit -a
 - ❖ git push

What is mrb?

- ❖ Multiple-repository build system, simplifies the building of multiple products pulled from different repositories
- ❖ *setup mrb*
- ❖ *mrb newDev -h ##* Will list lots of info about newDev command

mrb -h

Usage /products/larsoft/mrb/v1_04_05/bin/mrb (newDev | gitCheckout | svnCheckout | mrbsetenv | build | install | test | makePackage | mrbslp | zapBuild | newProduct | changelog | updateDepsCM | updateDepsPV | checkDeps | pullDeps | makeDeps) [-h for help]"

Tools (for help on tool, do "/products/larsoft/mrb/v1_04_05/bin/mrb <tool> -h")

newDev (n)	Start a new development area
gitCheckout (g)	Clone a git repository
svnCheckout (svn)	Checkout from a svn repository
build (b)	Run buildtool
install (i)	Run buildtool with install
test (t)	Run buildtool with tests
makePackage (mp)	Make distribution tarballs
zapBuild (z)	Delete everything in your build area
newProduct (p)	Create a new product from scratch
changelog (c)	Display a changelog for a package
updateDepsCM (uc)	Update the master CMakeLists.txt file
updateDepsPV (uv)	Update a product version in product_deps
updateSource	Update all svn or git code in MRB_SOURCE
makeDeps (md)	Build or update a header level dependency list
checkDeps (cd)	Check for missing build packages
pullDeps (pd)	Pull missing build packages into MRB_SOURCE

Aliases (we use aliases for these commands because they must be sourced)

mrbsetenv	Setup a development environment (source \$MRB_DIR/bin/mrbSetEnv)
mrbslp directory	Setup all products installed in the working localProducts_XXX (source \$MRB_DIR/bin/setup_local_products)

LArSoft versioning

- ❖ When you setup a version of LArSoft you do the following:
 - ❖ `setup larsoft v06_01_00 -q e10:prof`
- ❖ The version (06_01_00) has 3 parts;
 - ❖ 1st number is major version, increments slowly and only when there are big breaking changes eg moving to art v2 and ROOT6 (July 2016)
 - ❖ 2nd number is minor version, increments when new features such as data product members are added
 - ❖ 3rd number is patch number, this increments roughly
- ❖ The qualifier (e10:prof) has two parts
 - ❖ 1st number is the qualifier, it increments for newer versions of gcc
 - ❖ Seems to be cause for a major version?, for example v05 had e9.
 - ❖ 2nd number is the compiler, and there are two options
 - ❖ Prof – profiled, runs faster but less useful debugging
 - ❖ Debug – runs slower, but easier to debug.

How repositories talk to each other

- ❖ In a local checkout of a repository you have a file called `ups/product_deps` in here it will have a line which says which version of the given repository it calls itself.
- ❖ It will also say which version of both LArSoft and some key repositories it depends on.
- ❖ If you do not have these other repositories checked out then your code will depend on the code which was in develop when that version of the repository was made.
 - ❖ This means that to have the most up-to-date code you have to all the repositories checked out and continuously do git pull.
 - ❖ Obviously a bit daft to do (compiling would take ages and you'd just be pulling code all the time), hence a new release ~every week.
- ❖ If you have repositories checked out then you will use the code which is in your `srcs` directory not that which is in develop.

```
[php13tkw@dunegpvm08 srcs\ ]$ ls
CMakeLists.txt  dependency_list  dunetpc  duneutil  larreco
```


How to setup your local environment

- ❖ We now have a good enough idea about how stuff works to get our hands on some code!
- ❖ One question to consider though. Do you want to work on computers in Minnesota or at FNAL?
- ❖ Both will work exactly the same way but you have to do the initial setup each time you log differently.
- ❖ If FNAL

```
source /grid/fermiapp/products/dune/setup_dune.sh
```

- ❖ If Minnesota (well Sheffield)
 - ❖ This requires someone to have installed CVFMS onto a server locally somewhere.

```
source /cvmfs/fermilab.opensciencegrid.org/products/larsoft/setup
source /cvmfs/dune.opensciencegrid.org/products/dune/setup
```


How to setup your local environment 2

- ❖ Now we want to setup larsoft and get our repos.
- ❖ Check what is the latest version of larsoft
 - ❖ `ups list -aK+ larsoft`
 - ❖ This works for all repos. & art products eg GEANT4
- ❖ Setup larsoft of the desired version
 - ❖ `setup larsoft v06_01_00 -q e10:prof`
- ❖ Make a new directory for LArSoft
 - ❖ If using FNAL machines, DO NOT use your home area (AFS).
 - ❖ DO use your `/dune/app/users/USER/` area.
 - ❖ Be careful about making any soft links between AFS and `/dune/app/users/USER`. It can make grid submission awkward
 - ❖ `mkdir larDev`
 - ❖ `cd larDev`

How to setup your local environment 3

- ❖ You now want make a new development area.
 - ❖ `mrbs newDev`
 - ❖ This only works in an empty directory
- ❖ You will get an output saying you need to source something now and whenever you logon.
 - ❖ `source localProducts_XXXX/setup`
- ❖ You now want to get your repositories
 - ❖ `cd srcs (cd $MRBS_SOURCEDIR)`
 - ❖ `mrbs g dunetpc`
 - ❖ `mrbs g < any other repository your heart desires >`

How to setup your local environment 4

- ❖ You now want to build your code.
 - ❖ `cd ../build` (`cd $MRB_BUILDDIR`)
 - ❖ `mrbsenv`
 - ❖ `mrbi -j8`
- ❖ When you have made changes to your code and need to recompile:
 - ❖ `cd $MRB_BUILDDIR`
 - ❖ `make install -j 8`
- ❖ You then want to make sure that you are using your local products
 - ❖ `mrbslp`
- ❖ In the above commands the `-j X` tells the compiler how many cores to use.
- ❖ The `mrbi -j 8` command can be split into two commands if you want to do the building and installing separately.
 - ❖ `mrbbuild -j 8`
 - ❖ `make install`

How to setup your local environment 5

- ❖ Some pointers about building.
 - ❖ DUNE has a buildmachine (dunebuild01), it has 16 cores so is much faster for building.
 - ❖ Whenever you check out a new repo or add a new file you have to do
 - ❖ `cd $MRB_BUILDDIR`
 - ❖ `mrbsenv`
 - ❖ `mrbi -j 16`
 - ❖ There is a compiler called ninja which at least feels faster
 - ❖ `cd $MRB_BUILDDIR; mrbz; mrbsenv`
 - ❖ `setup ninja v1_6_0`
 - ❖ `mrbi -j 16 -generator ninja`
 - ❖ When using ninja the *make install -j8* command on the previous slide changes to
 - ❖ `ninja install -j 8`

What to do when you log back in

- ❖ Luckily you don't have to do this every time you login, you only have to do a small subset of the commands.
- ❖ Left – my script for FNAL
- ❖ Right – my script for Sheffield

```
#!/bin/sh

DIRECTORY=/dune/app/users/php13tkw/LarDevelop/

source /grid/fermiapp/products/dune/setup_dune.sh
echo 'Local software sourced'
echo 'Larsoft set up'
source ${DIRECTORY}/localProducts_larsoft_*/setup
echo 'Local products sourced'
cd ${DIRECTORY}/build_slf6.x86_64/
mrbsenv
mrbslp
mrbslp
cd ${DIRECTORY}/
echo

. /grid/fermiapp/products/common/etc/setups.sh
setup jobsub_client
setup ninja v1_6_0
```

```
#!/bin/sh

DIRECTORY=/home/warburton/LArSoft

source /cvmfs/fermilab.opensciencegrid.org/products/larsoft/setup
source /cvmfs/dune.opensciencegrid.org/products/dune/setup
setup mrb
setup ninja v1_6_0
export MRB_PROJECT=larsoft
echo 'Local software sourced'

source ${DIRECTORY}/localProducts_larsoft_*/setup
echo 'Local products sourced'

cd ${DIRECTORY}/build_slf6.x86_64/
mrbsenv
mrbslp
mrbslp

cd ${DIRECTORY}

export FW_SEARCH_PATH=./:${FW_SEARCH_PATH}
```


New LArSoft releases

- ❖ There is a new release ~ every week, so you making a new directory for each release would be pain! This means we will have to update our code to rely on the newest release when one comes out.
- ❖ First thing, log out and then log back in again!
- ❖ Setup LArSoft as we did previously
 - ❖ `source /grid/fermiapp/products/dune/setup_dune.sh`
 - ❖ `setup larsoft v06_01_01 -q e10:prof`
 - ❖ `cd larDev`
- ❖ We now want to make a new development within our current directory
 - ❖ `mrbs newDev -p`
 - ❖ The `-p` option tells mrbs to make a new localProducts using an existing src directory.
- ❖ Source the new localProducts
 - ❖ `source localProducts_XXXX/setup`

New LArSoft releases

- ❖ Now we want to update our repositories
 - ❖ `cd $MRB_SOURCEDIR / dunetpc`
 - ❖ `git checkout develop`
 - ❖ `git pull`
- ❖ If working on a feature branch, want to do two more commands
 - ❖ `git checkout feature / Karl_NewBranch`
 - ❖ `git merge develop`
 - ❖ `git push`
- ❖ Update other repositories such as larreco, larsim etc.
- ❖ Now, go to build directory and do a clean build.
 - ❖ `mrb z; mrbsetenv; mrb i -j16 --generator ninja`

Multiple builds

- ❖ It is possible to have multiple build areas (a debug and a prof) which depend on the same srcs directory.
 - ❖ Debug for testing, prof for running jobs.
- ❖ Clean login!
- ❖ Setup environment and desired LArSoft
 - ❖ `cd larDev`
 - ❖ `mrbs newDev -v v06_01_00 -q debug:e10 -T debug -f`
 - ❖ `-T` specifies name of new directory
 - ❖ `-f` specifies that you want to use existing srcs
- ❖ This makes new directory debug with a localProducts and build directory in it
- ❖ Source the new localProducts and build!
- ❖ Now, when you logon you have the choice of using either prof or debug, and they both use the same srcs directory.

Producers, Analyzers and Algs

- ❖ Producers and analyzers define modules which are ran at your desire when you run LArSoft
- ❖ Algorithms however just hold code, and are accessed by producers and analyzers.
 - ❖ Obviously advantageous to put quite general code in algs so that multiple modules can use the same code e.g. calorimetry calculations are in algs.
- ❖ A producer produces something, thus changing the event record eg hit reconstruction
- ❖ An analyzer just analyzes the data.
- ❖ There are also source modules but these are rarely used, and filters which are very useful though I have little experience using them.

Adding a new module

- ❖ At some point you will want to make a new module for your work.
- ❖ First step is to decide if it is a producer / analyzer i.e. does it add anything to the data record?
- ❖ Second step is decide which repo and subdirectory to put it in.
- ❖ You can make a brand new empty module
- ❖ OR, you can just copy an existing module, renaming it and changing the name of the class etc. (much easier).
- ❖ You then need to make sure it will get built, this is done by looking in the CMakeLists.txt file in that directory.
 - ❖ Hopefully this is no effort, as it should art_make which tells the compiler to build everything in the directory, but some directories don't do this yet...Then you need to either change it, or add it to the list.
- ❖ Now do a clean build and your module is ready for you to run.

Adding a new directory

- ❖ If you want to put your module in a new directory.
- ❖ Make your new directory.
- ❖ Add an extra `add_subdirectory(DirName)` line to the `CMakeLists.txt` in the repo.
- ❖ Example in `larreco / larreco`
- ❖ Make your new module, and copy a `CMakeLists.txt` into your new directory.
- ❖ Do a clean build and you're ready to go.

```
add_subdirectory(ClusterFinder)
add_subdirectory(EventFinder)
add_subdirectory(Genfit)
add_subdirectory(HitFinder)
add_subdirectory(RecoAlg)
add_subdirectory(ShowerFinder)
add_subdirectory(TrackFinder)
add_subdirectory(VertexFinder)
add_subdirectory(SpacePointFinder)
add_subdirectory(MCComp)
add_subdirectory(WireCell)
add_subdirectory(DirOfGamma)
```


What we've covered

- ❖ A quick overview of:
 - ❖ Git
 - ❖ LArSoft versioning and how to use mrb to get repositories
 - ❖ Setting up your local environment
 - ❖ Using multiple build areas
 - ❖ Adding new modules and directories